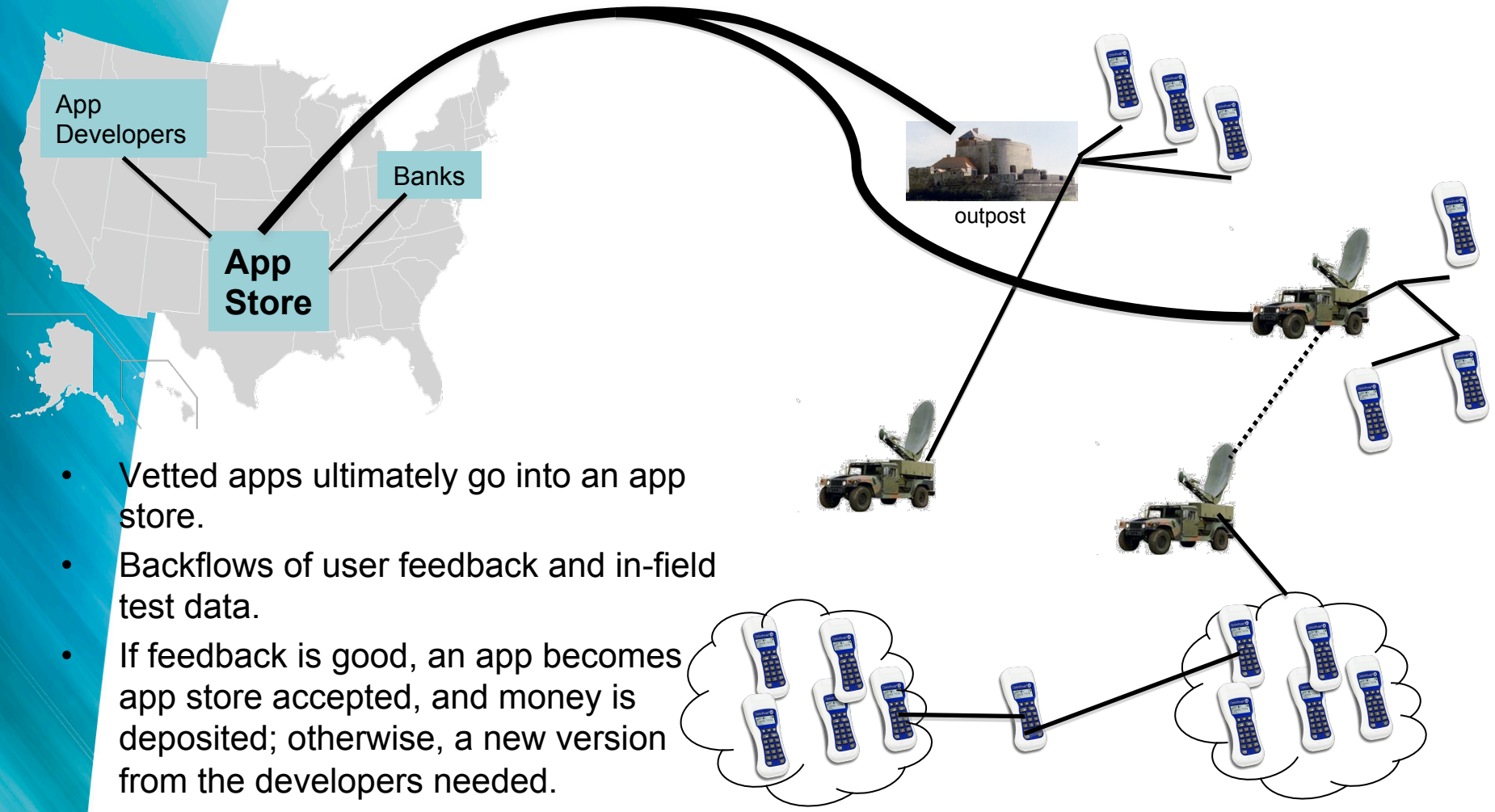
Vetting Applications

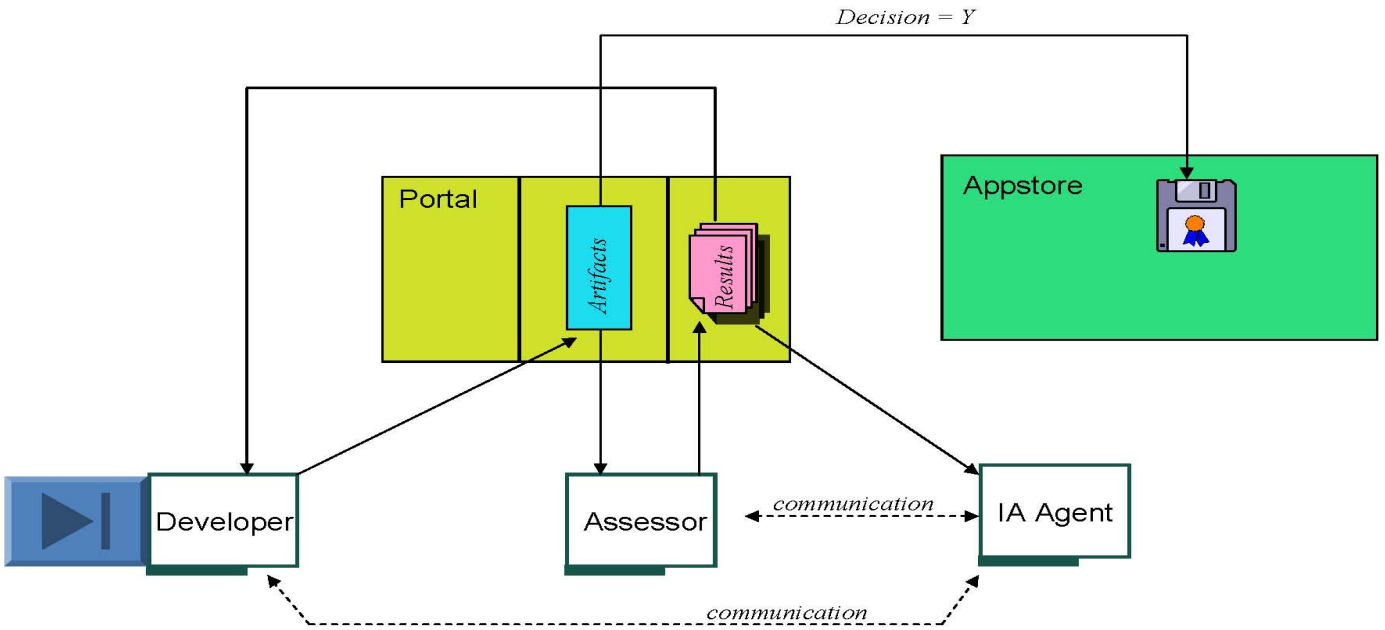
Jeff Voas & Angelos Stavrou
NIST
George Mason University

High-Level Project Overview

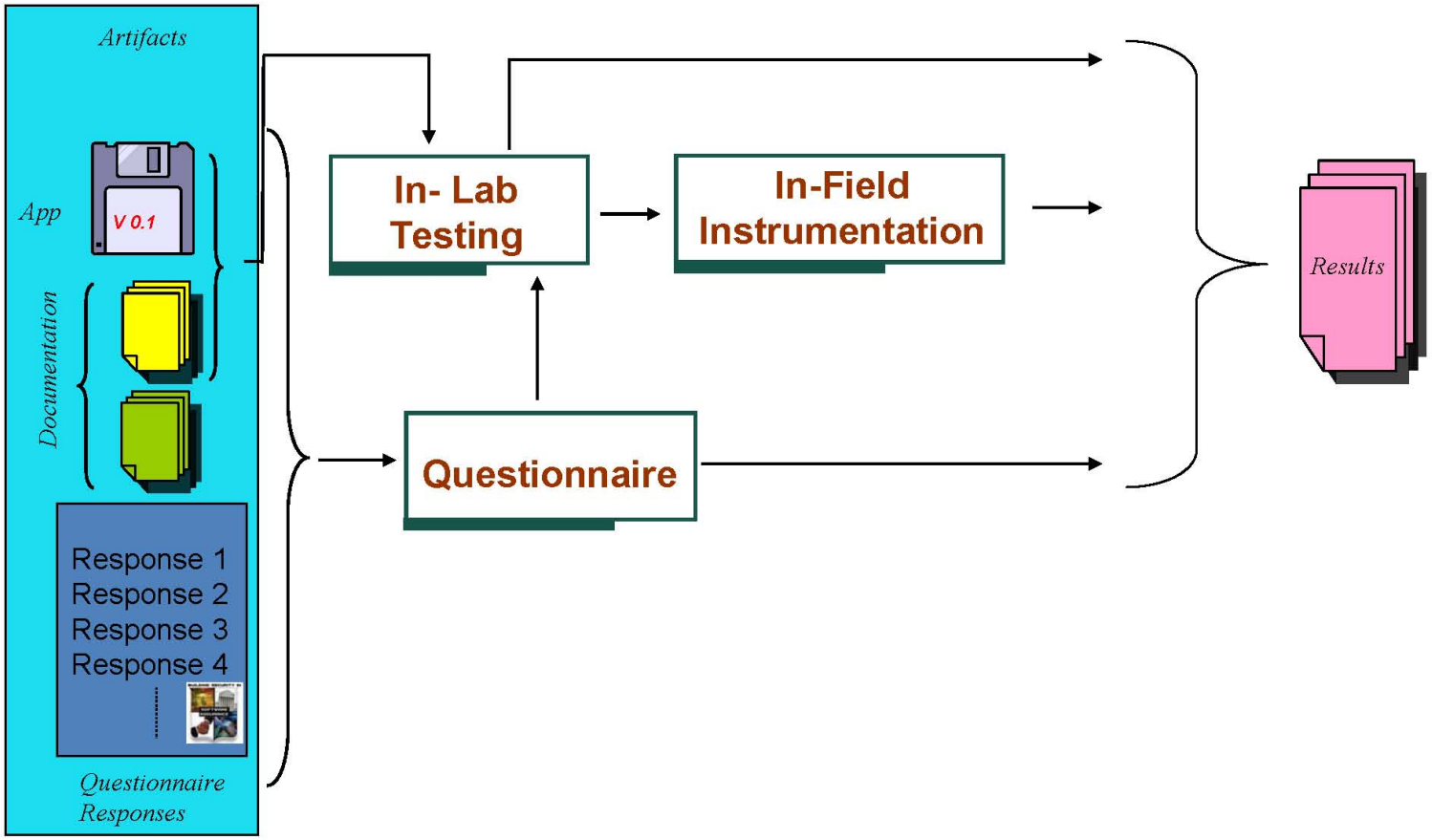


- Vetted apps ultimately go into an app store.
- Backflows of user feedback and in-field test data.
- If feedback is good, an app becomes app store accepted, and money is deposited; otherwise, a new version from the developers needed.

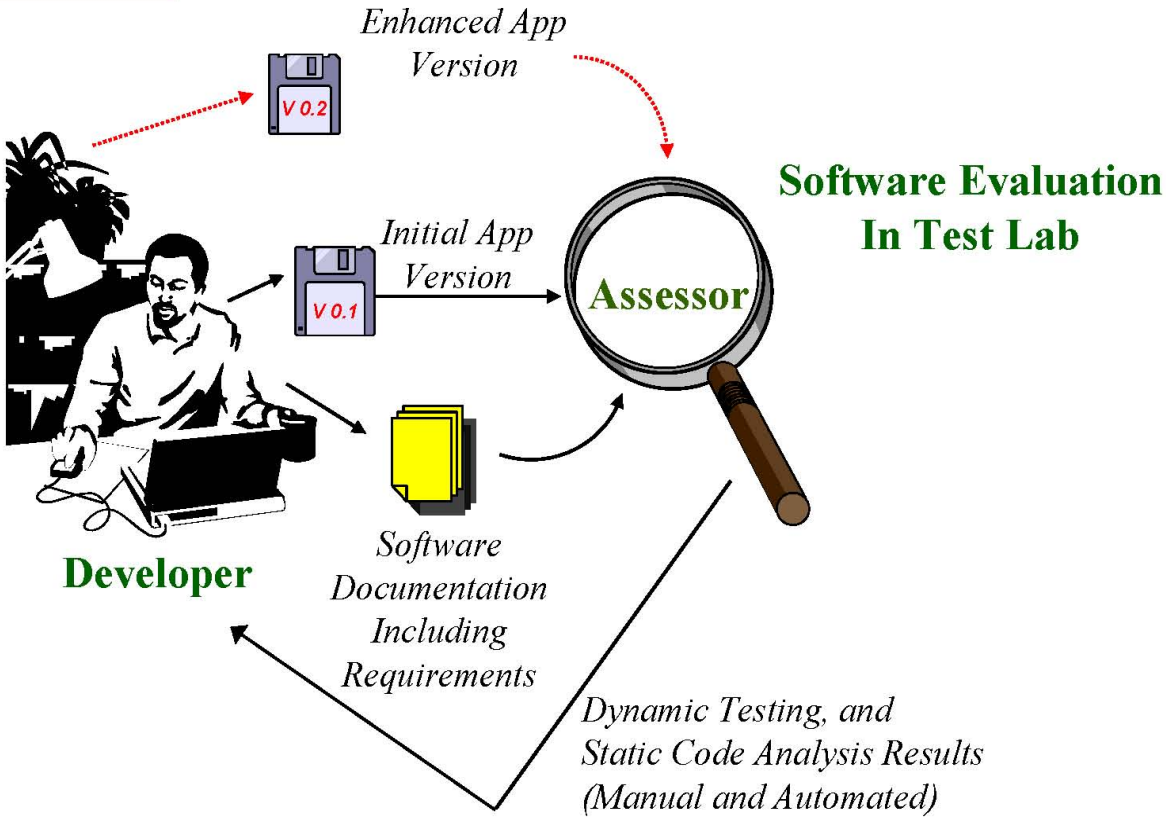
Application Vetting: Big Picture



Progression of Testing



In-Lab Testing Process



What about existing Analysis Tools?

- Commercial application testing tools cover regular, non-Android specific Bugs:
  - No Security Analysis of the Code Functionality
  - No Power Analysis of the Application components and code
  - No Profiling of the resource consumption of individual applications
  - Cannot Regulate/Deny the access and use of phone subsystems (Camera, Microphone, GPS..)
- Existing tools **do not cover Program Functionality**
  - We reveal the application capabilities and access

Application Testing Framework

Application Static Analysis does not cover Program Functionality

Fortify, Coverity, and other application testing tools cover regular, non-Android **specific Bugs:**

- No Security Analysis of the Code Functionality
- No Power Analysis of the Application components and code
- No **Profiling** of the resource consumption of individual applications
- **Cannot Regulate/Deny** the access and use of phone subsystems (Camera, Microphone, GPS..)

App Vetting & Control

- App Signing – Prevent unauthorized App Execution
  - Approved Apps are signed by the program designated approval authority
  - Only program signed Apps can be installed on the device
    - Customizations made to Android package framework
- App Analysis & Testing
  - All Apps are analyzed for malware and potential vulnerabilities
    - AV Scans
    - Vulnerability Scans (Fortify)
  - Expose hidden & unwanted functionality
    - Hidden in Native Libraries
    - Dynamic or obfuscated code
  - Permissions manifest reconciliation against code

Android Application Control

- Application Signing – Prevent unauthorized App Execution
  - Approved Apps are signed by the program designated approval authority
  - Only program signed Apps can be installed on the device
    - Customizations made to Android package framework
- Application Stress Testing
  - Measure Power Consumption
  - Identify Input Errors / Find UI bugs

Application Analysis Framework

- Android Specific Analysis includes analysis of the Application Security Manifest
  - Tailored to the Android Permission Model
- Verify if the requested permissions are warranted by the submitted code
  - Remove excessive permissions & enforce a tighter security model
- Regulate access to critical/restricted resources
  - Modifications on the Android Engine to enable dynamic policies
  - Control the underlying Dalvik engine to report absence/depletion of resources instead of lack of permissions

Application Policy Enforcement

Solution: Per Application Policy Enforcement

Provide Dalvik mechanisms to

- Enforce application Access & Capabilities
  - Tailored to specific Location or Time
  - Tailored to specific Mission
- Application can still be installed but deprived access to resources and data selectively

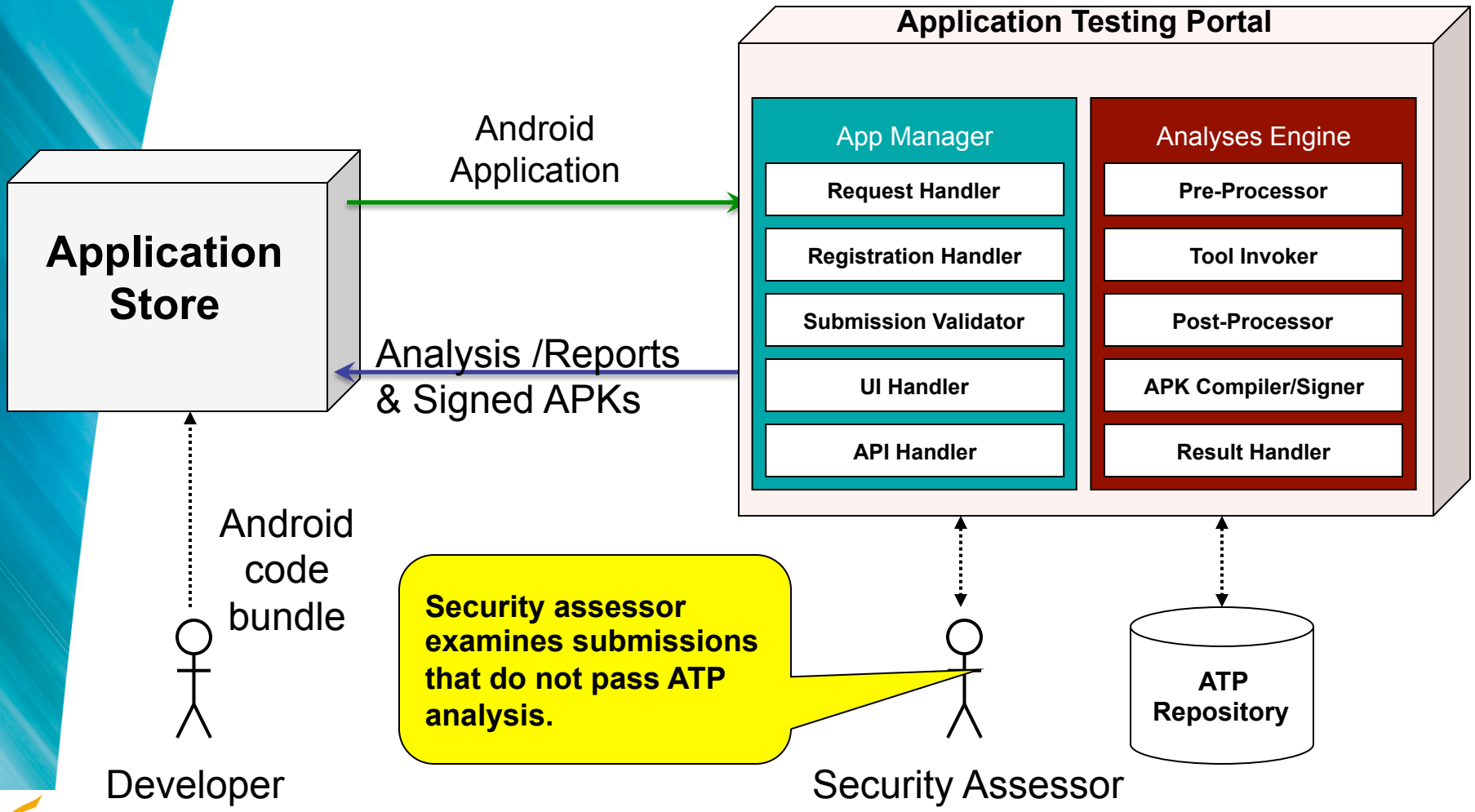
Policy Enforcement paired with Device Security can significantly reduce the risk of **Data Exfiltraction**

Power Metering Framework

- Design & Implement an accurate model for accounting and policing energy consumption
- Two-pronged approach
  - Meter the per-process CPU & Device utilization over time
  - Identify the relative impact of each device component on energy consumption
- Design an Android kernel subsystem to estimate energy
  - Meter energy consumption for each App/process
  - Use for characterizing application behavior
  - This behavior is Application dependent
  - Sometimes the behavior is also User dependent

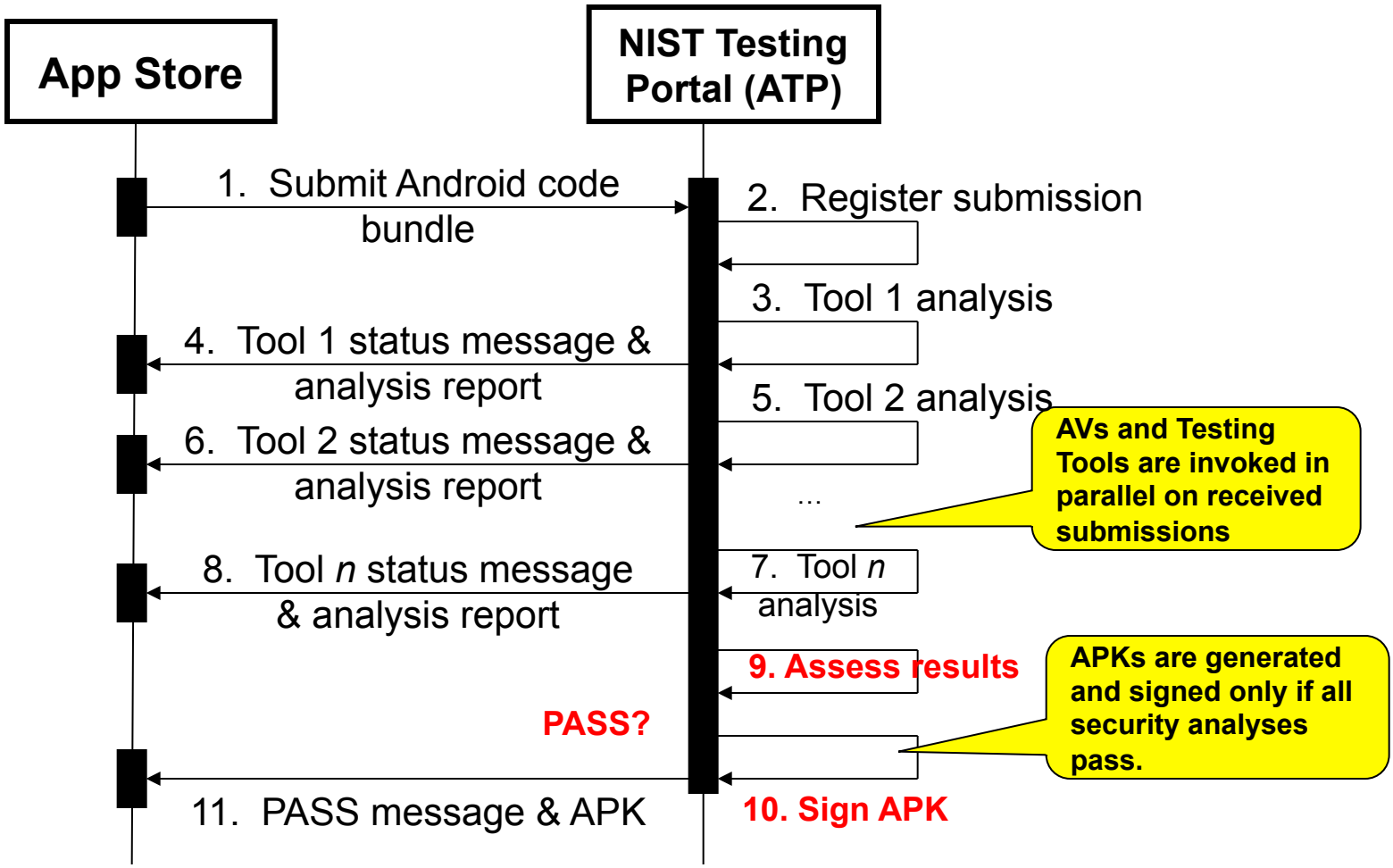
ATP Architecture

ATP analyzes Android code bundles and returns messages, analysis reports, and signed APKs

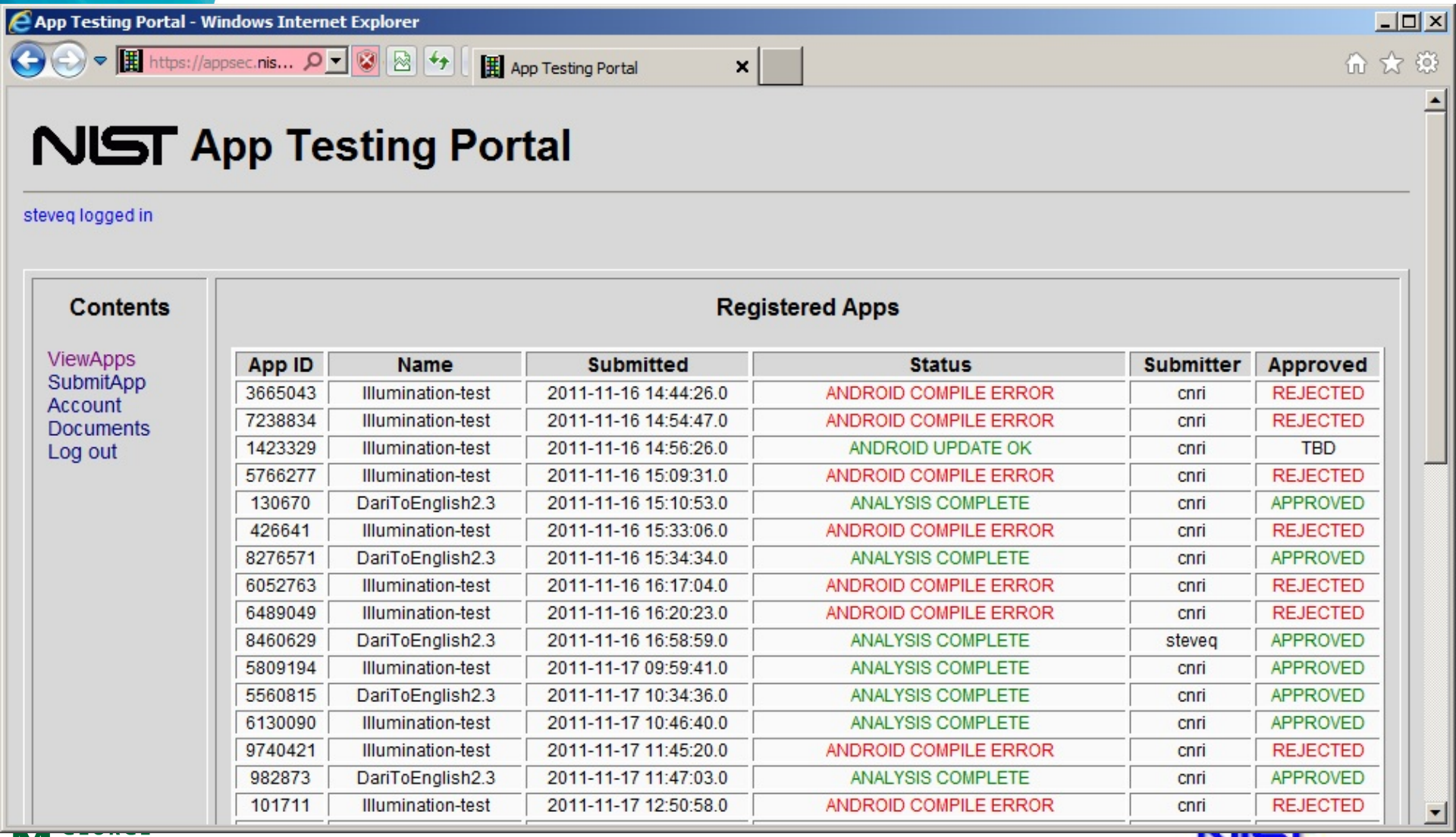


Mobilize-ATP Workflow (PASS Use-Case)

ATP applies Testing to Analyze Android code bundles



ATP Monitor


App Testing Portal - Windows Internet Explorer
https://appsec.nis...
App Testing Portal
NIST App Testing Portal
steveq logged in
Contents
ViewApps
SubmitApp
Account
Documents
Log out
Registered Apps
App ID | Name | Submitted | Status | Submitter | Approved
3665043 | Illumination-test | 2011-11-16 14:44:26.0 | ANDROID COMPILE ERROR | cnri | REJECTED
7238834 | Illumination-test | 2011-11-16 14:54:47.0 | ANDROID COMPILE ERROR | cnri | REJECTED
1423329 | Illumination-test | 2011-11-16 14:56:26.0 | ANDROID UPDATE OK | cnri | TBD
5766277 | Illumination-test | 2011-11-16 15:09:31.0 | ANDROID COMPILE ERROR | cnri | REJECTED
130670 | DariToEnglish2.3 | 2011-11-16 15:10:53.0 | ANALYSIS COMPLETE | cnri | APPROVED
426641 | Illumination-test | 2011-11-16 15:33:06.0 | ANDROID COMPILE ERROR | cnri | REJECTED
8276571 | DariToEnglish2.3 | 2011-11-16 15:34:34.0 | ANALYSIS COMPLETE | cnri | APPROVED
6052763 | Illumination-test | 2011-11-16 16:17:04.0 | ANDROID COMPILE ERROR | cnri | REJECTED
6489049 | Illumination-test | 2011-11-16 16:20:23.0 | ANDROID COMPILE ERROR | cnri | REJECTED
8460629 | DariToEnglish2.3 | 2011-11-16 16:58:59.0 | ANALYSIS COMPLETE | steveq | APPROVED
5809194 | Illumination-test | 2011-11-17 09:59:41.0 | ANALYSIS COMPLETE | cnri | APPROVED
5560815 | DariToEnglish2.3 | 2011-11-17 10:34:36.0 | ANALYSIS COMPLETE | cnri | APPROVED
6130090 | Illumination-test | 2011-11-17 10:46:40.0 | ANALYSIS COMPLETE | cnri | APPROVED
9740421 | Illumination-test | 2011-11-17 11:45:20.0 | ANDROID COMPILE ERROR | cnri | REJECTED
982873 | DariToEnglish2.3 | 2011-11-17 11:47:03.0 | ANALYSIS COMPLETE | cnri | APPROVED
101711 | Illumination-test | 2011-11-17 12:50:58.0 | ANDROID COMPILE ERROR | cnri | REJECTED

Defense in-Depth: Multiple Levels of Security

- Application Vetting & Testing
- Device Lock-down and Encryption of ALL Data and Communications
- Enforcement of Security Policies in the Android Framework
- Second-level Defenses placed in the Android Linux Kernel
  - Prevent Attacks that bypass Android Security Framework
  - Android has Inherited some (if not all) of the Linux Vulnerabilities
  - **Java Native Interface to Linux Libraries a potential Avenue for Exploitation**

Conclusions

Assuring the Secure Operation of Smart Devices has a wide-range of requirements

- Application Testing
  - Static & Dynamic
  - In-Field Instrumentation
  - Power Behavior Metering & Policing
- Physical Device Security
  - Lock-Down of the Device I/O (USB, WiFi, etc.)
  - Encryption of Data both on the Phone & Network
  - Securing Provisioning Process